On the convergence of hybrid decomposition methods for SVM training

S. Lucidi, L. Palagi* and A. Risi, M. Sciandrone**

*Università di Roma “La Sapienza”
Dipartimento di Informatica e Sistemistica “A. Ruberti”
Via Buonarroti 12 - 00185 Roma - Italy

**Istituto di Analisi dei Sistemi ed Informatica del CNR “A. Ruberti”
Viale Manzoni 30 - 00185 Roma - Italy

e-mail (Stefano Lucidi): lucidi@dis.uniroma1.it
e-mail (Laura Palagi): palagi@dis.uniroma1.it
e-mail (Arnaldo Risi): sciandro@iasi.rm.cnr.it
e-mail (Marco Sciandrone): sciandro@iasi.rm.cnr.it

ABSTRACT

Support Vector Machines (SVM) is a widely adopted technique both for classification and regression problems. Training of SVM requires to solve a linearly constrained convex quadratic problem. In real applications the number of training data may be very huge and the Hessian matrix cannot be stored. In order to take into account this issue a common strategy consists in using decomposition algorithms which at each iteration operate only on a small subset of variables, usually referred to as the working set. The convergence properties of a decomposition method strongly depend on the rule for selecting the working set. On the other hand training time can be significantly reduced by using a caching technique, that allocates some memory space to store the recently used columns of the Hessian matrix. In order to take into account both theoretical requirements and computational issues related to a caching technique, we propose a hybrid decomposition algorithm model embedding the most violating pair rule and possibly any other working set selection rule. We prove the global convergence of the proposed hybrid algorithm model.

1This work was supported in part by MIUR, Research program PRIN 2005 Problemi e metodi innovativi nell’ottimizzazione non lineare, Italy.
Furthermore, we present two specific practical realizations of the general hybrid model where a caching strategy can be advantageously exploited.

**Keywords.** Support Vector Machines, large scale quadratic optimization, caching technique, working set selection, maximum violating pair, convergent hybrid schemes.
1 Introduction

Given a training set of input-target pairs \((x^i, y^i), \ i = 1, \ldots, l\), with \(x^i \in \mathbb{R}^m\), and \(y^i \in \{-1, 1\}\), the SVM classification technique requires the solution of the following convex quadratic programming problem

\[
\begin{align*}
\min_{\alpha} \quad & f(\alpha) = \frac{1}{2} \alpha^T Q \alpha - e^T \alpha \\
\text{s.t.} \quad & y^T \alpha = 0 \\
& 0 \leq \alpha \leq C e,
\end{align*}
\]

where \(\alpha \in \mathbb{R}^l\), \(Q\) is a \(l \times l\) positive semidefinite matrix, \(e \in \mathbb{R}^l\) is the vector of all ones, \(y \in \{-1, 1\}^l\) and \(C\) is a positive scalar. The generic element \(q_{ij}\) of the matrix \(Q\) is given by \(y^i y^j K(x^i, x^j)\), where \(K(x, z) = \phi(x)^T \phi(z)\) is the kernel function related to the nonlinear function \(\phi\) that maps the data from the input space into the feature space.

We assume that the number \(l\) of training data is huge and the Hessian matrix \(Q\) cannot be fully stored so that standard methods for Quadratic Programming cannot be used. Hence the adopted strategy to solve the SVM problem is usually based on the decomposition of the original problem into a sequence of smaller subproblems obtained by fixing subsets of variables.

Formally, starting from a feasible point \(\alpha^k\), at the current iteration \(k\) the index sets of the variables is partitioned into two subsets \(W\) (with cardinality \(|W| = q \geq 2\)), usually called working set, and \(\overline{W} = \{1, \ldots, l\} \setminus W\) (for notational convenience we omit in some cases the dependence on \(k\)). The variables corresponding to \(\overline{W}\) are unchanged at the new iteration, while the variables corresponding to \(W\) are set to a solution \(\alpha^*_W\) of the subproblem

\[
\begin{align*}
\min_{\alpha_W, \alpha^k_W} \quad & f(\alpha_W, \alpha^k_W) \\
& y^T_W \alpha_W = -y^T_W \alpha^k_W \\
& 0 \leq \alpha_W \leq C e_W
\end{align*}
\]

Thus the new iterate is \(\alpha^{k+1} = (\alpha^{k+1}_W, \alpha^{k+1}_\overline{W}) = (\alpha^*_W, \alpha^k_\overline{W})\).

The selection rule of the working set strongly affects both the speed of the algorithm and its convergence properties.

In computational terms the most expensive step at each iteration of a decomposition method is the evaluation of the kernel to compute the columns of the Hessian matrix, corresponding to the index in the working set \(W\), not stored in memory. Note that the \(q\) columns corresponding
to the working set $W$ must be necessarily used at least for the gradient updating, being

$$
\nabla f(\alpha^{k+1}) = \nabla f(\alpha^k) + Q(\alpha^{k+1} - \alpha^k)
= \nabla f(\alpha^k) + \sum_{i \in W} Q_i(\alpha_i^{k+1} - \alpha_i^k),
$$

where $\nabla f(\alpha) = Q\alpha - e$ and where $Q_i$ is the $i-$th column of the Hessian matrix $Q$.

To reduce the computational time a commonly adopted strategy is based on the use of a caching technique that allocates some memory space (the cache) to store the recently used columns of $Q$ thus avoiding in some cases the recomputation of these columns. Hence, to minimize the number of kernel evaluations and to reduce the computational time it is convenient to select working sets containing as much as possible elements corresponding to columns stored in the cache memory.

However, to guarantee the global convergence of a decomposition method the working set selection cannot be completely arbitrary but must satisfy suitable rules, for instance, those based on the “maximum violation” of the optimality conditions (see, e.g., [5], [6], and [7]). On the other hand, “maximum violation” rules are not designed to fully exploit the information of the matrix cache. Therefore the study of decomposition methods specifically designed to couple both the theoretical aspects and an efficient use of a caching strategy is of interest and has motivated this work.

Recently an interesting hybrid decomposition approach has been studied in [3]. The proposed hybrid scheme combines the maximal violation rule with a working set selection rule using second order information from the matrix cache. The number of variables entering the working set at each iteration is two, and in special situations it is imposed to select the most violating pair, namely the pair corresponding to the maximal violation of the optimality conditions. This occurs when both variables indexed by the preceding working set are not “sufficiently far” from the bounds. The hybrid decomposition algorithm is globally convergent. The convergence analysis strongly depends on the specific selection rules adopted in the algorithm. The computational experiments show the effectiveness of the hybrid approach thanks to its efficient usage of the matrix cache.

In this work we define a general convergent hybrid decomposition model which embeds the “maximum violation” selection rule to guarantee theoretical properties and permits to adopt any arbitrary choice (even in the dimension) of the working set provided that a suitable condition, easy to be tested, is satisfied. Due to its generality, the resulting
scheme may lead to several practical algorithms where the degree of freedom introduced in the scheme could be fruitfully employed by means of a caching technique. As examples, we present two specific algorithms derived by the general decomposition framework. The focus of the paper is mainly theoretical and computational issues will not be analyzed.

The paper is organized as follows. In Section II we describe a commonly adopted decomposition scheme, the Most Violating Pair Algorithm, related to our hybrid approach. In Section III we present the general hybrid decomposition scheme. Two specific algorithms derived by the general model are described in Section IV. The convergence analysis is performed in Section V. Finally Section VI contains some concluding remarks.

2 The Most Violating Pair (MVP) Algorithm

In this section we recall briefly a decomposition algorithm scheme based on the “maximum violation” of the optimality conditions, which play an important role in the definition of working set selection rules. In fact, the popular and efficient convergent decomposition methods SVMlight [5] and LIBSVM [1], [2] use the information deriving from the violation of the optimality conditions.

Throughout the paper, we denote by \( F \) the feasible set of Problem (1), namely
\[
F = \{ \alpha \in \mathbb{R}^l : y^T \alpha = 0, \ 0 \leq \alpha \leq Ce \},
\]
and by \( \nabla f = Q\alpha - e \) the gradient of \( f \).

Since the feasible set \( F \) is compact, Problem (1) admits solution. Moreover, as \( f \) is convex and the constraints are linear, a feasible point \( \alpha^* \) is a solution of Problem (1) if and only if the Karush-Kuhn-Tucker (KKT) conditions are satisfied.

In [6] KKT conditions have been written in a quite compact form that has been used to define a convergent decomposition algorithm. In particular, let us introduce in a feasible point \( \alpha \) the index sets:
\[
R(\alpha) = \{ i : \alpha_i < C, \ y_i = 1 \} \cup \{ i : 0 < \alpha_i, \ y_i = -1 \},
\]
\[
S(\alpha) = \{ i : \alpha_i < C, \ y_i = -1 \} \cup \{ i : 0 < \alpha_i, \ y_i = 1 \}.
\]

We note that \( R(\alpha) \cap S(\alpha) = \{ i : 0 < \alpha_i < C \} \) and \( R(\alpha) \cup S(\alpha) = \{1, \ldots, \ell\} \).

We report the KKT conditions in the following proposition [6].
**Proposition 1 (Optimality conditions)** A point $\alpha^* \in \mathcal{F}$ is a solution of Problem (1) if and only if

$$
\max_{h \in R(\alpha^*)} -y_h \nabla f(\alpha^*)_h \leq \min_{h \in S(\alpha^*)} -y_h \nabla f(\alpha^*)_h
$$

(3)

Given a feasible point $\alpha$ which is not a solution of Problem (1), a pair $i \in R(\alpha)$, $j \in S(\alpha)$ such that

$$
-y_i \nabla f(\alpha)_i > -y_j \nabla f(\alpha)_j
$$

is said a violating pair.

A very simple decomposition method for SVM is the Sequential Minimal Optimization (SMO) algorithm [10]. SMO-type algorithms select at each iteration working set of dimension exactly two, so that the updated point can be analytically computed, and this eliminates the need to use an optimization software.

It has been shown in [4] that SMO-type algorithms have strict decrease of the objective function if and only if the working set is a violating pair. However, the use of generic violating pairs as working sets is not sufficient to guarantee convergence properties of the sequence generated by a decomposition algorithm.

A modified version of SMO has been proposed in [6], where the two indices selected in the working set are those corresponding to the “maximal violation” of the KKT conditions.

More specifically, given again a feasible point $\alpha$ which is not a solution of Problem (1), let us define

$$
I(\alpha) = \left\{ i : i \in \arg \max_{h \in R(\alpha)} -y_h \nabla f(\alpha)_h \right\},
$$

$$
J(\alpha) = \left\{ j : j \in \arg \min_{h \in S(\alpha)} -y_h \nabla f(\alpha)_h \right\}.
$$

Taking into account the KKT conditions as stated in (3), a pair $i \in I(\alpha)$, $j \in J(\alpha)$ most violate the optimality conditions and therefore it is said a maximal violating pair.

A SMO-type algorithm using maximal violating pairs as working sets is usually called Most Violating Pair (MVP) algorithm.

MVP algorithm is related to the hybrid decomposition approach we propose and it is formally described below.
Algorithm MVP

Data. A feasible point $\alpha^0$.

Inizialization. Set $k = 0$.

While (stopping criterion not satisfied)

1. Select a most violating pair
   \[ i(k) \in I(\alpha^k), \quad j(k) \in J(\alpha^k). \]

2. Set $W = \{i(k), j(k)\}$, compute the solution $\alpha^*_W$ of
   \[
   \min f(\alpha_W, \alpha^k_W) \\
   y^T_W \alpha_W = -y^T_W \alpha^k_W \\
   0 \leq \alpha_W \leq C e_W,
   \]
   and set $\alpha^{k+1} = (\alpha^*_W, \alpha^k_W)$.

3. Set $k = k + 1$.

end while

Return $\alpha^* = \alpha^k$

MVP algorithm can in turn be viewed as a special case of the SVMlight algorithm [5], which is based on a specific procedure for choosing the $q$ elements of the working set, being $q$ any even number. The convergence properties of SVMlight algorithm have been proved in [7] and in [8] under suitable convexity assumptions.

In the next section we define a general hybrid scheme embedding the MVP strategy.

3 A general hybrid algorithm model (HMVP)

In our hybrid scheme, under certain conditions related to the MVP algorithm, it is admitted the possibility of arbitrarily choosing the new updated point. The basic idea is to use, as a reference value, the value of the objective function obtained by the MVP strategy. Any point that allows a sufficient decrease with respect to this value can be accepted as
the new iterate. The general hybrid model can be formally described as follows.

**Algorithm Model HMVP**

**Data.** A feasible point \( \alpha^0 \), a real \( \sigma > 0 \), an integer \( p \geq 1 \).

**Initialization.** Set \( k = 0 \).

**While (stopping criterion not satisfied)**

1. Select a most violating pair
   
   \[ i(k) \in I(\alpha^k), \quad j(k) \in J(\alpha^k). \]

2. Set \( W = \{i(k), j(k)\} \), compute the solution \( \alpha_W^* \) of
   
   \[
   \min f(\alpha_W, \alpha_{Wk}^k) \\
   y_W^T \alpha_W = -y_W^T \alpha_{Wk}^k \\
   0 \leq \alpha_W \leq C e_W,
   \]

   and set \( \hat{\alpha}^k = (\alpha_W^*, \alpha_{Wk}^k) \).

3. If
   
   \[ 0 < \hat{\alpha}_{i(k)} < C \quad \text{and} \quad 0 < \hat{\alpha}_{j(k)} < C \quad (4) \]

   then find \( \alpha^{k+1} \) s.t.
   
   \[
   f(\alpha^{k+1}) \leq f(\hat{\alpha}^k) - \sigma \| \alpha^{k+1} - \hat{\alpha}^k \|^p \quad (5)
   \]

   else
   
   set \( \alpha^{k+1} = \hat{\alpha}^k \)

   end if

4. Set \( k = k + 1 \).

end while

**Return** \( \alpha^* = \alpha^k \)

Starting from the current point \( \alpha^k \), the “reference” point \( \hat{\alpha}^k \) is computed according to the MVP strategy. If the two updated variables are not at the bounds (see condition (4)) then any arbitrary point which provides a
sufficient reduction of the function value can be chosen as the new iterate. Note that checking condition (4) does not involve the computation of the gradient and hence avoids the relevant additional cost due to (possible) restore of columns of the Hessian matrix not stored in the cache memory.

Algorithm HMVP allows in principle the use of working sets of arbitrary dimension provided that (4) is satisfied. The convergence properties of Algorithm HMVP will be analyzed in the next section.

We observe that a convergent hybrid algorithm, the Hybrid Maximum-Gain (HMG) Algorithm, has been proposed in [3]. It is based on the switching from a working set rule using second order information (taken from cached kernel matrix entries) to the MVP rule. Algorithm HMG is a SMO-type method, and the switching condition for applying the second order selection rule, instead of the MVP rule, is that both variables indexed by the previous working set are “sufficiently far” from the bounds.

4 Two practical realizations of HMVP

As examples, we present two specific realizations of Algorithm HMVP where the information of the matrix cache can be advantageously exploited. In both the algorithms, we use the letter $W$ to indicate the MVP working set and the letter $w$ to indicate the alternative working set that can be used to define the updated point.

Algorithm H1

Data. A feasible point $\alpha^0$.

Initialization. Set $k = 0$.

While (stopping criterion not satisfied)

1. Select a most violating pair $i(k) \in I(\alpha^k), j(k) \in J(\alpha^k)$.

2. Set $W = \{i(k), j(k)\}$, compute the solution $\alpha^*_W$ of

$$\min f(\alpha_W, \alpha^k_W)$$

$$y^T_W \alpha_W = -y^T_W \alpha^k_W$$

$$0 \leq \alpha_W \leq C e_W,$$

and set $\hat{\alpha}^k = (\alpha^*_W, \alpha^k_W)$.
3. If
   \[ 0 < \hat{\alpha}_i(k) < C \quad \text{and} \quad 0 < \hat{\alpha}_j(k) < C \]
   then
   3a) Select \( s(k) \) and \( t(k) \) such that \( Q_s \) and \( Q_t \) are in the cache memory.
   3b) Set \( w = \{s(k), t(k)\} \), compute the solution \( \alpha_{w}^{k+1} \) of
   \[
   \min f(\alpha_w, \hat{\alpha}_{\overline{w}}^{k})
   \]
   \[
   y_{w}^T \alpha_w = -y_{\overline{w}}^T \hat{\alpha}_{\overline{w}}^{k}
   \]
   \[
   0 \leq \alpha_w \leq C e_w,
   \]
   and set \( \alpha^{k+1} = (\alpha_w^{k+1}, \hat{\alpha}_{\overline{w}}^{k}) \).
   else
   set \( \alpha^{k+1} = \hat{\alpha}^k \)
   end if
4. Set \( k = k + 1 \).

end while

Return \( \alpha^* = \alpha^k \)

The strategy of Algorithm H1 is that of performing, when possible, a two-step minimization with respect to two different pairs of variables. The former corresponds to the maximum violating pair, the latter corresponds to a pair of columns stored in the cache memory and must be related to a violating pair at the reference point \( \hat{\alpha}^k \) (otherwise we would necessarily have \( \alpha^{k+1} = \hat{\alpha}^k \)). This implies that in practice the gradient \( \nabla f(\hat{\alpha}^k) \) must be computed to define the working set \( w = \{s(k), t(k)\} \).

Summarizing, at each iteration, Algorithm H1 moves at least two variables and at most four variables. In any case, the updated point can be analytically determined, furthermore at most two columns of the Hessian matrix (those corresponding to the maximum violating pair) must be possibly recomputed.

The proof of Corollary 1, stated in the next section, shows that the instructions at Step 3 of Algorithm H1 imply that condition (5) of Algorithm HVMP holds, and hence that Algorithm H1 is a specific realization of Algorithm HVMP.

As second example derived by the general hybrid decomposition scheme we present the following algorithm.
Algorithm H2

Data. A feasible point $\alpha^0$.

Inizialization. Set $k = 0$.

While (stopping criterion not satisfied)

1. Select a most violating pair
   
   \[ i(k) \in I(\alpha^k), \quad j(k) \in J(\alpha^k). \]

2. Set $W = \{i(k), j(k)\}$ and compute the solution $\alpha^*_W$ of
   \[
   \min f(\alpha_W, \alpha^k_W) \\
   y^T_W \alpha_W = -y^T_W \alpha^k_W \\
   0 \leq \alpha_W \leq C e_W,
   \]
   and set $\hat{\alpha}^k = (\alpha^*_W, \alpha^k_W)$.

3. If $0 < \hat{\alpha}_{i(k)} < C$ and $0 < \hat{\alpha}_{j(k)} < C$
   
   then
   
   3a) Select $h_i, i = 1, \ldots, p$, such that $Q_{h_i}, i = 1, \ldots, p$, are in the cache memory.
   
   3b) Set

   \[ w = \{(s(k), t(k))\} \cup \{h_i, i = 1, \ldots, p\}, \]

   compute the solution $\alpha^{k+1}_w$ of
   \[
   \min f(\alpha_w, \hat{\alpha}^k_w) \\
   y^T_w \alpha_w = -y^T_w \hat{\alpha}^k_w \\
   0 \leq \alpha_w \leq C e_w,
   \]
   and set $\alpha^{k+1} = (\alpha^{k+1}_w, \hat{\alpha}^k_w)$.

   else
   
   set $\alpha^{k+1} = \hat{\alpha}^k$

   end if
4. Set $k = k + 1$.

\textbf{end while}

\textbf{Return} $\alpha^* = \alpha^k$

Note that Algorithm H2 admits the possibility of updating, at each iteration, more than two variables, and it fruitfully exploits the matrix cache. Differently from Algorithm H1, in Algorithm H2 it is not necessary to compute the gradient at the reference point $\hat{\alpha}^k$. As the dimension of the working set $w$ is greater than two, we can think to apply an iterative method for solving subproblem (7). Thus, condition (5) can be used in principle as termination criterion during the inner iterations of the iterative solver, for instance, whenever an approximate solution of (7) has not yet been attained but the reduction of the objective function becomes not satisfactory.

The proof of Corollary 2 stated in the next section shows that the instructions at Step 3 of Algorithm H2 imply that condition (5) of Algorithm HVMP holds, and hence that Algorithm H2 is a specific realization of Algorithm HVMP.

A strategy similar to that of H2 scheme has been proposed in [11], where theoretical aspects have not been analyzed but computational issues related to working set selections rule have been deeply investigated in a gradient projection-based decomposition framework. The effectiveness of the proposed strategy has been shown by numerical experiments on large benchmark problems.

5 Convergence analysis

In this section we prove the convergence properties of Algorithm HMVP. To this aim we preliminarily state some technical results not specifically related to Algorithm HMVP.

Given a feasible point $\alpha$, the set of the feasible directions at $\alpha$ is the cone

$$D(\alpha) = \{d \in R^l : y^T d = 0, \quad d_i \geq 0, \forall i : \alpha_i = 0, \quad \text{and} \quad d_i \leq 0, \forall i : \alpha_i = C\}.$$ 

The following two propositions have been proved in [9].

\textbf{Proposition 2} Let $\alpha$ be a feasible point. For each pair $i \in R(\alpha)$ and $j \in S(\alpha)$, the direction $d \in R^l$ such that

$$d_i = y_i, \quad d_j = -y_j, \quad d_h = 0 \quad \text{for} \quad h \neq i, j$$

12
is a feasible direction at \( \alpha \), i.e. \( d \in D(\alpha) \).

**Proposition 3** Let \( \{\alpha^k\} \) be a sequence of feasible points convergent to a point \( \bar{\alpha} \). Then for sufficiently large values of \( k \) we have

\[
R(\bar{\alpha}) \subseteq R(\alpha^k) \quad \text{and} \quad S(\bar{\alpha}) \subseteq S(\alpha^k).
\]

Now we prove the following result.

**Proposition 4** Assume that \( Q \) is a symmetric positive semidefinite matrix, and let \( \alpha^k \) be a feasible point for Problem (1). Let \( W = \{i(k), j(k)\} \), let \( \alpha^*_W \) be the solution of subproblem (2), and set

\[
\hat{\alpha}^k = (\alpha^*_W, \alpha^k_W).
\]

If

\[
i(k) \in R(\hat{\alpha}^k) \quad \text{and} \quad j(k) \in S(\hat{\alpha}^k)
\]  

then we have

\[
y_{i(k)} \nabla f(\hat{\alpha}^k)_{i(k)} - y_{j(k)} \nabla f(\hat{\alpha}^k)_{j(k)} \geq 0.
\]  

**Proof.** Assume by contradiction that (9) does not hold, i.e., that

\[
y_{i(k)} \nabla f(\alpha^*_W, \alpha^k_W)_{i(k)} - y_{j(k)} \nabla f(\alpha^*_W, \alpha^k_W)_{j(k)} < 0. \tag{10}
\]

From (8) and Proposition 2 we get that the direction \( d^k \in R^l \) such that

\[
d^k_h = \begin{cases} 
  y_{i(k)} & \text{if } h = i(k) \\
  -y_{j(k)} & \text{if } h = j(k) \\
  0 & \text{otherwise}
\end{cases} \tag{11}
\]

is a feasible direction at \( \hat{\alpha}^k = (\alpha^*_W, \alpha^k_W) \), furthermore, (10) implies that \( d^k \) is also a descent direction at \( \hat{\alpha}^k \). Therefore, for sufficiently small values of \( \beta \) we can write

\[
\hat{\alpha}^k + \beta d^k = (\alpha^*_W + \beta d^k_W, \alpha^k_W) \in F,
\]

and

\[
f(\alpha^*_W + \beta d^k_W, \alpha^k_W) < f(\alpha^*_W, \alpha^k_W),
\]

but this contradicts the fact that \( \alpha^*_W \) is a solution of subproblem (2).

Finally the following result holds (see [7] and [8]).

**Proposition 5** Assume that
either $Q$ is a symmetric positive definite matrix and $W$ is an arbitrary working set,

or $Q$ is a symmetric positive semidefinite matrix and $W$ is a working set of dimension two.

Let $\alpha^k$ be a feasible point for Problem (1) and let $\alpha^*_W$ be the solution of subproblem (2). Then there exists $\gamma > 0$ such that

$$f(\alpha^*_W, \alpha^k_W) \leq f(\alpha^k) - \gamma \|\alpha^*_W - \alpha^k\|^2.$$ 

The next proposition concerns theoretical properties used later for proving the global convergence of Algorithm HMVP.

**Proposition 6** Assume that Algorithm HMVP does not terminate and let $\{\alpha^k\}$ be the sequence generated by it. Then we have

$$\lim_{k \to \infty} \|\hat{\alpha}^k - \alpha^k\| = 0. \quad (12)$$

and

$$\lim_{k \to \infty} \|\alpha^{k+1} - \alpha^k\| = 0.. \quad (13)$$

**Proof.** First of all we have (see Proposition 5)

$$f(\hat{\alpha}^k) \leq f(\alpha^k) - \gamma \|\hat{\alpha}^k - \alpha^k\|^2, \quad (14)$$

and, by the instructions of the algorithm, we can write

$$f(\alpha^{k+1}) \leq f(\hat{\alpha}^k) \leq f(\alpha^k) - \gamma \|\hat{\alpha}^k - \alpha^k\|^2. \quad (15)$$

As $\alpha^k$ belongs to the compact set $\mathcal{F}$ for all $k$ and $f$ is bounded below on $\mathcal{F}$, from (15) we get

$$\lim_{k \to \infty} f(\alpha^k) = \lim_{k \to \infty} f(\hat{\alpha}^k) = \bar{f}. \quad (16)$$

From (14) and (16) we obtain that (12) is proved.

Let $K_1 \subseteq \{0, 1, \ldots\}$ be such that for all $k \in K_1$ condition (4) is satisfied so that

$$f(\alpha^{k+1}) \leq f(\hat{\alpha}^k) - \sigma \|\alpha^{k+1} - \hat{\alpha}^k\|^p \quad \forall k \in K_1. \quad (17)$$

Let $K_2$ be such that $K_1 \cup K_2 = \{0, 1, \ldots\}$ and $K_1 \cap K_2 = \emptyset$. From the instructions at Step 3, for all $k \in K_2$ we have $\alpha^{k+1} = \hat{\alpha}^k$ and hence, using (14), we can write

$$f(\alpha^{k+1}) \leq f(\alpha^k) - \gamma \|\alpha^{k+1} - \alpha^k\|^2 \quad \forall k \in K_2. \quad (18)$$
If $K_1$ is an infinity subset then from (16) and (17) we obtain
\[
\lim_{k \to \infty, k \in K_1} \| \hat{\alpha}^k - \alpha^{k+1} \| = 0,
\]
which implies, together with (12), that
\[
\lim_{k \to \infty, k \in K_1} \| \alpha^{k+1} - \alpha^k \| = 0.
\]
If $K_2$ is an infinity subset then using (16) and (18) we obtain
\[
\lim_{k \to \infty, k \in K_2} \| \alpha^{k+1} - \alpha^k \| = 0,
\]
and this concludes the proof.

Now we are ready to prove the asymptotic convergence of Algorithm HMVB.

**Proposition 7** Assume that Algorithm HMVP does not terminate, and let $\{\alpha^k\}$ be the sequence generated by it. Then, every limit point of $\{\alpha^k\}$ is a solution of Problem (1).

**Proof.** Let $\bar{\alpha}$ be any limit point of a subsequence of $\{\alpha^k\}$, i.e. there exists an infinite subset $K \subseteq \{0, 1, \ldots\}$ such that $\alpha^k \to \bar{\alpha}$ for $k \in K$, $k \to \infty$. By contradiction, let us assume that $\bar{\alpha}$ is not a KKT point for Problem (1). By Proposition 1 there exists at least a pair $(i, j) \in R(\bar{\alpha}) \times S(\bar{\alpha})$ such that:
\[
-y_i \nabla f(\bar{\alpha})_i > -y_j \nabla f(\bar{\alpha})_j.
\]
(19)

For a pair $(i(k), j(k)) \in R(\alpha) \times S(\alpha)$, let us introduce the notation $d_{i(k), j(k)} \in R^l$ as
\[
d_{i(k), j(k)}^h = \begin{cases} y_{i(k)} & \text{if } h = i(k) \\ -y_{j(k)} & \text{if } h = j(k) \\ 0 & \text{otherwise.} \end{cases}
\]

The proof is divided into two parts.

(a) First assume that there exists a number $M$ such that for all $k \in K$ and $k$ sufficiently large we can find an index $m(k)$, with $0 \leq m(k) \leq M$, for which we have
\[
i(k + m(k)) \in R(\hat{\alpha}^{k+m(k)}) \quad \text{and} \quad j(k + m(k)) \in S(\hat{\alpha}^{k+m(k)}).
\]

Note that the satisfaction of condition (4) at Step 3 implies that the above condition holds. From Proposition 4 it follows
\[
\nabla f(\hat{\alpha}^{k+m(k)})^T d_{i(k+m(k)), j(k+m(k))} \geq 0.
\]
(20)
As \( \alpha^k \to \bar{\alpha} \) for \( k \in K \) and \( k \to \infty \), and \( m(k) \leq M \), from (13) we obtain
\[
\lim_{k \to \infty, k \in K} \alpha^{k+m(k)} = \bar{\alpha}.
\] (21)

Furthermore, from (12) we get
\[
\lim_{k \to \infty, k \in K} \hat{\alpha}^k = \bar{\alpha}.
\] (22)

Since \( i(k + m(k)) \) and \( j(k + m(k)) \) belong to the finite set \( \{1, \ldots, l\} \), we can extract a further subset of \( K \), that we relabel again with \( K \), such that
\[
i(k + m(k)) = \hat{i} \quad j(k + m(k)) = \hat{j} \quad \text{for all } k \in K.
\]

Taking limits in (20), using (22) and recalling the continuity of the gradient, we obtain
\[
\nabla f(\bar{\alpha})^T d_{\hat{i}, \hat{j}} \geq 0.
\] (23)

From (21) and Proposition 3, since \( i \in R(\bar{\alpha}) \) and \( j \in S(\bar{\alpha}) \), we have that \( i \in R(\alpha^{k+m(k)}) \) and \( j \in S(\alpha^{k+m(k)}) \) for \( k \in K \) and \( k \) sufficiently large. Therefore, recalling that \( \hat{i} \in I(\alpha^{k+m(k)}) \) implies
\[
-y_i \nabla f(\alpha^{k+m(k)})_{\hat{i}} \geq -y_{\hat{i}} \nabla f(\alpha^{k+m(k)})_{i}
\]
and \( \hat{j} \in J(\alpha^{k+m(k)}) \) implies
\[
-y_j \nabla f(\alpha^{k+m(k)})_{\hat{j}} \leq -y_{\hat{j}} \nabla f(\alpha^{k+m(k)})_{j},
\]
we can write
\[
\nabla f(\alpha^{k+m(k)})^T d_{\hat{i}, \hat{j}} \leq \nabla f(\alpha^{k+m(k)})^T d_{i, j}.
\] (24)

Taking limits in (24), using (21) and recalling (23) we can write
\[
0 \leq \nabla f(\bar{\alpha})^T d_{\hat{i}, \hat{j}} \leq \nabla f(\bar{\alpha})^T d_{i, j},
\]
which contradicts (19).

(b) Since the assumptions stated in case (a) do not hold, it follows that for each integer \( M \geq 0 \) there exists an infinite subset \( K_M \subseteq K \) such that
\[
i(k + m) \notin R(\hat{\alpha}^{k+m}) \quad \text{or} \quad j(k + m) \notin S(\hat{\alpha}^{k+m})
\]
for all \( k \in K_M \) and for all \( m \in [0, M] \).

Thus, setting \( M = 2l \) and relabelling (if necessary) the subset \( K \) we can assume that for all \( k \in K \) and for all \( m \) such that \( 0 \leq m \leq 2l \) we
have necessarily that the condition (4) at Step 3 does not hold and hence that
\[ \alpha^{k+m+1} = \hat{\alpha}^{k+m}. \] (25)
Therefore, we must have
\[ i(k + m) \in R(\alpha^{k+m}) \quad \text{and} \quad j(k + m) \in S(\alpha^{k+m}) \]
and either
\[ i(k + m) \notin R(\alpha^{k+m+1}) \] (26)
or
\[ j(k + m) \notin S(\alpha^{k+m+1}). \] (27)
Thus there exists a partition of \( \{0, \ldots, 2l\} \) into two subsets \( \Gamma_1, \Gamma_2 \) such that for any \( m \in \Gamma_1 \), (26) holds and for any \( m \in \Gamma_2 \), (27) holds. For simplicity and without loss of generality, we consider only the case that \( |\Gamma_1| \geq l \), and hence that for all \( m \in \Gamma_1 \)
\[ i(k + m) \in R(\alpha^{k+m}) \quad \text{and} \quad i(k + m) \notin R(\alpha^{k+m+1}). \] (28)
From (28), as \( i(k + m) \in \{1, \ldots, l\} \) for all \( m \), we must have
\[ i(k + h(k)) = i(k + n(k)), \quad \text{with} \quad 0 \leq h(k) < n(k) \leq 2l, \]
and hence we can extract a further subset of \( K \) (that we relabel again \( K \)) such that for all \( k \in K \) we can write
\[ i(k + h(k)) = i(k + n(k)) = \hat{i}, \quad \text{with} \quad 0 \leq h(k) < n(k) \leq 2l. \]
Then, we can define a subset \( K_1 \) such that, for all \( k_i \in K_1 \),
\[ i(k_i) = i(k_{i+1}) = \hat{i}, \quad \text{with} \quad k_i < k_{i+1} \leq k_i + 2l, \] (29)
and we have
\[ \lim_{k_i \to \infty, k_i \in K_1} \alpha^{k_i} = \bar{\alpha}. \] (30)
From (25) we get, for all \( k_i \in K_1 \),
\[ \alpha^{k_i+m} = \hat{\alpha}^{k_i+m-1}, \quad m = 1, \ldots, 2l. \] (31)
We can write
\[ \hat{i} \in R(\alpha^{k_i}) \quad \text{and} \quad \hat{i} \notin R(\alpha^{k_i+1}) \quad \text{and} \quad \hat{i} \in R(\alpha^{k_i+1}), \]
that means that index \( \hat{i} \) must have been inserted in the working set of Step 2 and the corresponding component of the current point has
been modified by the optimization process (again of Step 2) between the iterates $k_i + 1$ and $k_i + 1 \leq k_i + l$.

Thus, for all $k_i \in K_1$, an index $p(k_i)$, with $k_i < p(k_i) \leq k_i + 1 \leq k_i + 2l$, exists such that $\hat{i} \notin R(\alpha^{p(k_i)})$ and

$$\hat{i} \in S(\alpha^{p(k_i)}) \text{ and } \hat{i} \in W^{p(k_i)}.$$  \hspace{1cm} (32)

As $p(k_i) - k_i \leq 2l$, recalling (30) and Proposition 6, we can write

$$\lim_{k_i \to \infty, k_i \in K_1} \alpha^{p(k_i)} = \bar{\alpha}. \hspace{1cm} (33)$$

Now we show that

$$-y_i \nabla f(\alpha \hat{i}_i) \geq -y_i \nabla f(\alpha \hat{i}), $$. \hspace{1cm} (34)

Indeed if this were not true, namely if

$$-y_i \nabla f(\alpha \hat{i}_i) > -y_i \nabla f(\alpha \hat{i}),$$

by the continuity of the gradient we would have for $k_i \in K_1$ and $k_i$ sufficiently large:

$$-y_i \nabla f(\alpha^k) \hat{i}_i > -y_i \nabla f(\alpha^k) \hat{i},$$

that in turns implies that $\hat{i} \notin I(\alpha^k)$ and hence that $i(k_i) \neq \hat{i}$ for $k_i \in K_1$ and sufficiently large which contradicts (29). Since (34) holds, using (19) we get

$$-y_i \nabla f(\alpha \hat{i}_i) > -y_i \nabla f(\alpha \hat{i}),$$

so that, recalling (33) and the continuity of the gradient we can write for $k_i \in K_1$ and $k_i$ sufficiently large

$$-y_i \nabla f(\alpha^{p(k_i)}) \hat{i}_i > -y_i \nabla f(\alpha^{p(k_i)}) \hat{i}. $$  \hspace{1cm} (35)

By (33) and Proposition 3, as $j \in S(\bar{\alpha})$, for $k_i \in K_1$ and $k_i$ sufficiently large we have that $j \in S(\alpha^{p(k_i)})$. Therefore (35) and the working set selection rule at Step 2 imply that $\hat{i} \notin W^{p(k_i)}$, but this contradicts (32).

We terminate the section by proving the global convergence of algorithms H1 and H2. To this aim it is sufficient to prove that each algorithm is a specific realization of Algorithm HVMP, namely that the instructions at Step 3 of each algorithm imply that condition (5) holds.
Corollary 1 Assume that $Q$ is a symmetric positive semidefinite matrix, that Algorithm H1 does not terminate and let $\{\alpha^k\}$ be the sequence generated by it. Then, every limit point of $\{\alpha^k\}$ is a solution of Problem (1).

Proof. From (6) and Proposition 5 we get that condition (5) holds with $\sigma = \gamma$ and $p = 2$, so that Algorithm H1 is a specific realization Algorithm HVMP and hence the thesis follows from Proposition 7.

Corollary 2 Assume that $Q$ is a symmetric positive definite matrix, that Algorithm H2 does not terminate and let $\{\alpha^k\}$ be the sequence generated by it. Then, every limit point of $\{\alpha^k\}$ is a solution of Problem (1).

Proof. Since $i(k), j(k) \in w$, and the reference point $\hat{\alpha}^k$ differs from $\alpha^k$ only in the two components defined by $i(k)$ and $j(k)$, we have that $\alpha_{\omega}^k = \hat{\alpha}_{\omega}^k$, and hence that subproblem (7) is

$$\min f(\alpha_{\omega}, \hat{\alpha}_{\omega}^k)$$

$$y_{\omega}^T \alpha_{\omega} = -y_{\omega}^T \hat{\alpha}_{\omega}^k$$

$$0 \leq \alpha_{\omega} \leq C e_{\omega},$$

so that from Proposition 5 we get that condition (5) holds with $\sigma = \gamma$ and $p = 2$. Therefore, Algorithm H2 is a specific realization Algorithm HVMP and hence the thesis follows from Proposition 7.

6 Conclusions

The aim of the work was that of defining a general decomposition algorithm for SVM training with guaranteed convergence properties and suitable degrees of freedom potentially useful from a computational point of view. To this aim, we have proposed a general hybrid decomposition method and its asymptotic convergence has been proved under the assumption that the quadratic objective function is convex. Although the focus of the work was theoretical, we have also presented two specific simple algorithms where the degrees of freedom of the general scheme can be fruitfully exploited by means of a caching strategy.

Finally we remark that the approach and the convergence analysis can be extended to the non convex case by employing proximal point techniques as in [9].
References


